Part-3-Data-Operation

December 21, 2021

1 Data Operations

Using the above data CSV file, we now gain insights.

```
[ ]: # Import libraries
     import pandas as pd
     import numpy as np

     import pickle

     import matplotlib.pyplot as plt
     import seaborn as sns
     %matplotlib inline

     from collections import Counter

     #Import Warnings
     import warnings
     warnings.filterwarnings("ignore")
```

1.1 Load Data

loading the game data with all the its columns.

```
[ ]: # Load games data
     gamesdata = pd.read_csv('gamesdata.csv', index_col = 0)
     gamesdata.head()
```

```
[ ]:           publisher                                            genres  \
     0         Kotoshiro  ['Action', 'Casual', 'Indie', 'Simulation', 'S…
     1  Making Fun, Inc.       ['Free to Play', 'Indie', 'RPG', 'Strategy']
     2      Poolians.com  ['Casual', 'Free to Play', 'Indie', 'Simulatio…
     3                                  ['Action', 'Adventure', 'Casual']
     4               NaN                                               NaN

                    app_name                    title  \
     0      Lost Summoner Kitty      Lost Summoner Kitty
     1                Ironbound                Ironbound
```

```
2  Real Pool 3D - Poolians  Real Pool 3D - Poolians
3                    2222                     2222
4           Log Challenge                      NaN

                                                url release_date  \
0  http://store.steampowered.com/app/761140/Lost_…   2018-01-04
1  http://store.steampowered.com/app/643980/Ironb…   2018-01-04
2  http://store.steampowered.com/app/670290/Real_…   2017-07-24
3     http://store.steampowered.com/app/767400/2222/   2017-12-07
4  http://store.steampowered.com/app/773570/Log_C…          NaN

                                                tags  discount_price  \
0  ['Strategy', 'Action', 'Indie', 'Casual', 'Sim…            4.49
1  ['Free to Play', 'Strategy', 'Indie', 'RPG', '…             NaN
2  ['Free to Play', 'Simulation', 'Sports', 'Casu…             NaN
3                  ['Action', 'Adventure', 'Casual']            0.83
4            ['Action', 'Indie', 'Casual', 'Sports']            1.79

                                       reviews_url  \
0  http://steamcommunity.com/app/761140/reviews/?…
1  http://steamcommunity.com/app/643980/reviews/?…
2  http://steamcommunity.com/app/670290/reviews/?…
3  http://steamcommunity.com/app/767400/reviews/?…
4  http://steamcommunity.com/app/773570/reviews/?…

                                               specs         price  \
0                                  ['Single-player']          4.99
1  ['Single-player', 'Multi-player', 'Online Mult…  Free To Play
2  ['Single-player', 'Multi-player', 'Online Mult…  Free to Play
3                                  ['Single-player']          0.99
4  ['Single-player', 'Full controller support', '…          2.99

   early_access        id          developer        sentiment  metascore
0         False  761140.0          Kotoshiro              NaN        NaN
1         False  643980.0  Secret Level SRL  Mostly Positive        NaN
2         False  670290.0      Poolians.com  Mostly Positive        NaN
3         False  767400.0                               NaN        NaN
4         False  773570.0                NaN              NaN        NaN
```

We also load the `mergeddata.csv` file which has a row for each user-item interaction.

[ ]:
```
# Load merged data
mergeddata = pd.read_csv('mergeddata.csv', index_col = 0)
mergeddata.head()
```

[ ]:
```
   uid  id  owned publisher      genres        app_name          title  \
0    0  10    1.0     Valve  ['Action']  Counter-Strike  Counter-Strike
1    1  10    1.0     Valve  ['Action']  Counter-Strike  Counter-Strike
```

```
2    3  10    1.0     Valve  ['Action']  Counter-Strike  Counter-Strike
3    4  10    1.0     Valve  ['Action']  Counter-Strike  Counter-Strike
4   10  10    1.0     Valve  ['Action']  Counter-Strike  Counter-Strike

                                             url release_date  \
0  http://store.steampowered.com/app/10/CounterSt…   2000-11-01
1  http://store.steampowered.com/app/10/CounterSt…   2000-11-01
2  http://store.steampowered.com/app/10/CounterSt…   2000-11-01
3  http://store.steampowered.com/app/10/CounterSt…   2000-11-01
4  http://store.steampowered.com/app/10/CounterSt…   2000-11-01

                                            tags  discount_price  \
0  ['Action', 'FPS', 'Multiplayer', 'Shooter', 'C…             NaN
1  ['Action', 'FPS', 'Multiplayer', 'Shooter', 'C…             NaN
2  ['Action', 'FPS', 'Multiplayer', 'Shooter', 'C…             NaN
3  ['Action', 'FPS', 'Multiplayer', 'Shooter', 'C…             NaN
4  ['Action', 'FPS', 'Multiplayer', 'Shooter', 'C…             NaN

                                     reviews_url  \
0  http://steamcommunity.com/app/10/reviews/?brow…
1  http://steamcommunity.com/app/10/reviews/?brow…
2  http://steamcommunity.com/app/10/reviews/?brow…
3  http://steamcommunity.com/app/10/reviews/?brow…
4  http://steamcommunity.com/app/10/reviews/?brow…

                                       specs price  early_access developer  \
0  ['Multi-player', 'Valve Anti-Cheat enabled']  9.99         False     Valve
1  ['Multi-player', 'Valve Anti-Cheat enabled']  9.99         False     Valve
2  ['Multi-player', 'Valve Anti-Cheat enabled']  9.99         False     Valve
3  ['Multi-player', 'Valve Anti-Cheat enabled']  9.99         False     Valve
4  ['Multi-player', 'Valve Anti-Cheat enabled']  9.99         False     Valve

                sentiment  metascore
0  Overwhelmingly Positive       88.0
1  Overwhelmingly Positive       88.0
2  Overwhelmingly Positive       88.0
3  Overwhelmingly Positive       88.0
4  Overwhelmingly Positive       88.0
```

And finally we load the `numgames.csv` file which just lists the number of games owned for each user.

[ ]:
```
# Load numgames data
numgames = pd.read_csv('numgames.csv', index_col = 0)
numgames.head()
```

```
[ ]:              user_id  items_count
     0  76561197970982479          277
     1            js41637          888
     2          evcentric          137
     3        Riot-Punch           328
     4              doctr          541
```

1.2 Exploration

1.2.1 User interaction data

```
[ ]: mergeddata['id'].nunique()
```

```
[ ]: 8171
```

```
[ ]: mergeddata['uid'].nunique()
```

```
[ ]: 8769
```

1.2.2 Release date

```
[ ]: # Select entries where release date is not null
     data = gamesdata[gamesdata['release_date'].notnull()]
```

```
[ ]: # Describe feature
     data['release_date'].describe()
```

```
[ ]: count           30068
     unique           3582
     top        2012-10-16
     freq              100
     Name: release_date, dtype: object
```

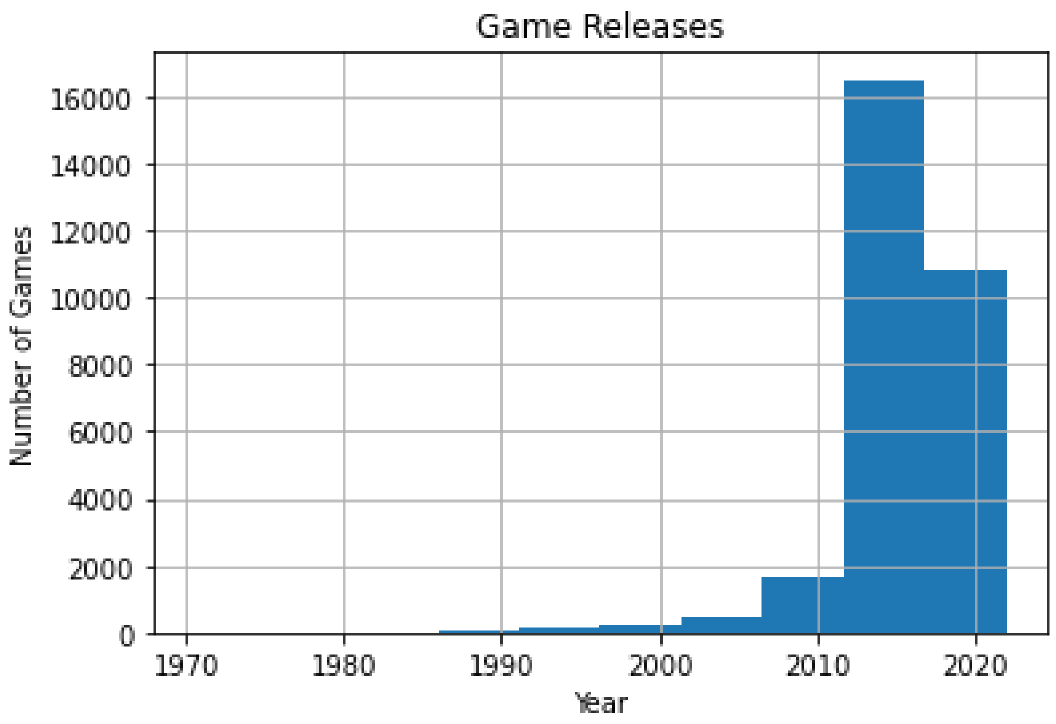
We note that there are 3582 unique values. We want to convert the type to Datetime instead of object.

```
[ ]: # Replace strings which are not of the format xxxx-xx-xx with None
     data['release_date'] = data['release_date'].map(lambda x : x if x[-3] ==␣
      ↪'-'else None)

     # Select entries where release date is not null
     data = data[data['release_date'].notnull()]

     # Convert to DateTime
     data['release_date'] = pd.to_datetime(data['release_date'])

     # Check
     data['release_date'].describe()
```

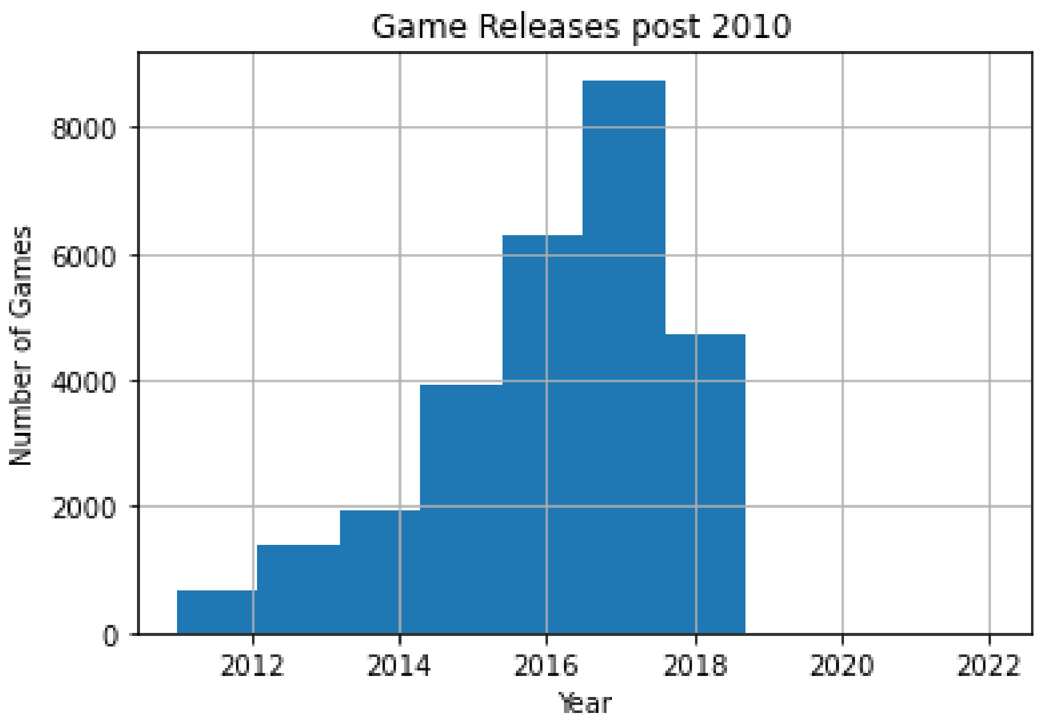
```
[ ]: count                      29783
     unique                      3457
     top        2012-10-16 00:00:00
     freq                         100
     first      1970-07-15 00:00:00
     last       2021-12-31 00:00:00
     Name: release_date, dtype: object
```

We see that our data contains games ranging from 1970 up to predicted release date of December 2021.

```
[ ]: # Plot histogram of release date feat
     data['release_date'].hist()
     plt.title('Game Releases')
     plt.ylabel('Number of Games')
     plt.xlabel('Year')
     plt.show()
```

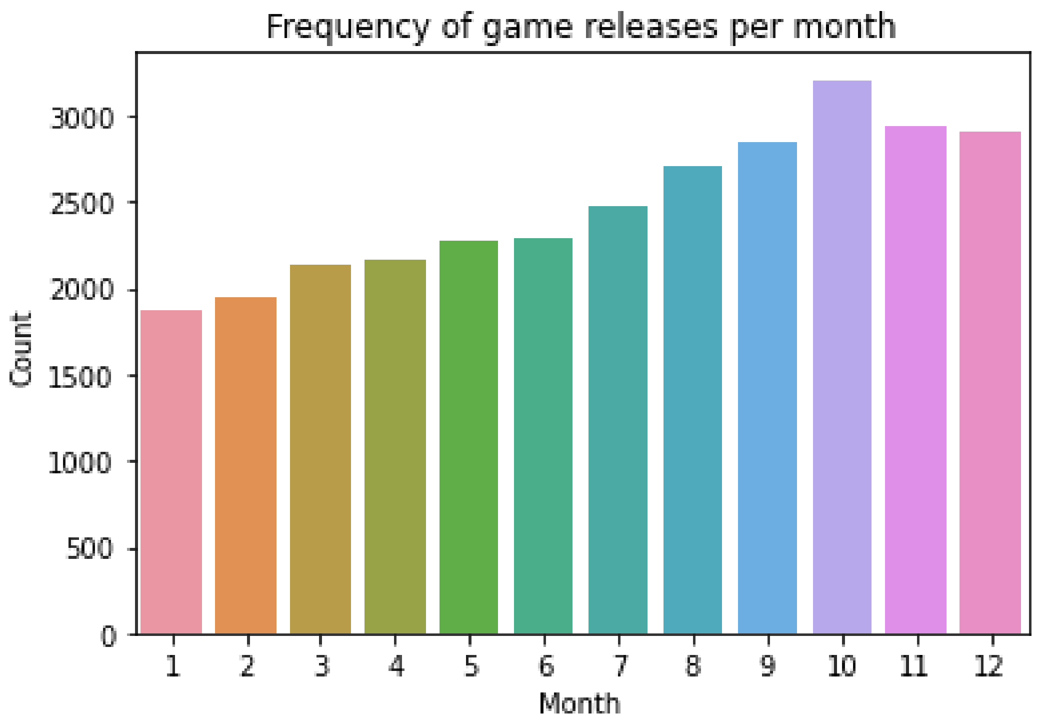


```
[ ]: # Focus on post 2010
     recentgames = data[data['release_date'].dt.year > 2010]

     recentgames['release_date'].hist()
     plt.title('Game Releases post 2010')
     plt.ylabel('Number of Games')
plt.xlabel('Year')
plt.show()
```

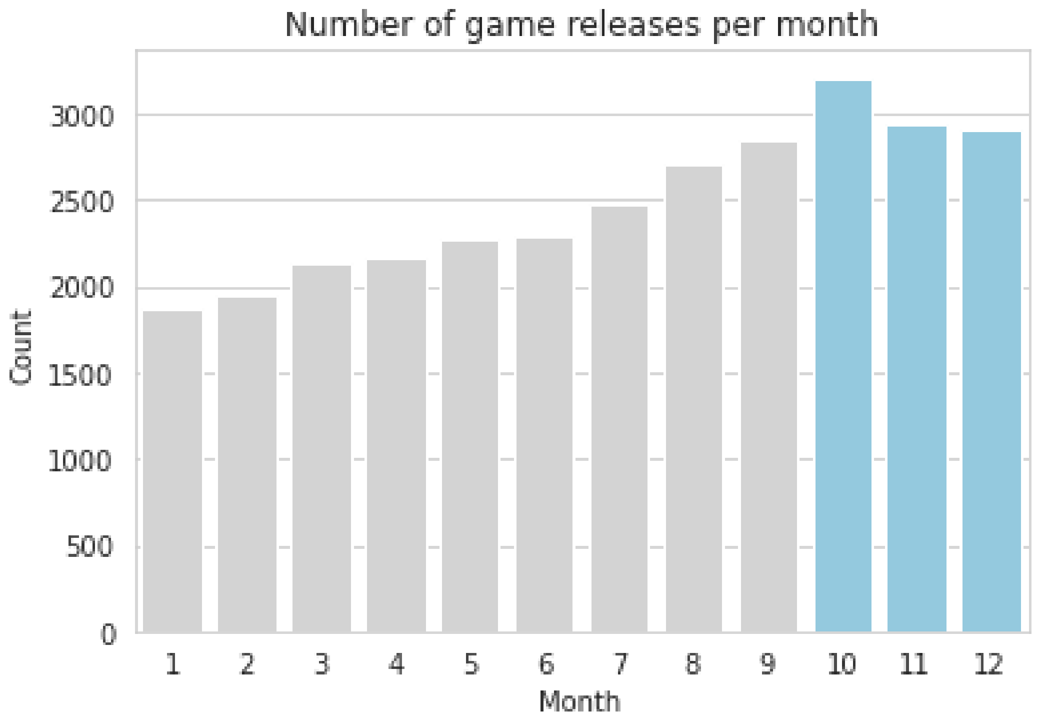


Let's see which months are most popular for new releases.

[ ]:
```
# Create month feature
data['release_month'] = data['release_date'].dt.month

# Plot countplot using Seaborn
sns.countplot(x = data['release_month'], data = data)
plt.title('Frequency of game releases per month')
plt.xlabel('Month')
plt.ylabel('Count')
plt.show()
```



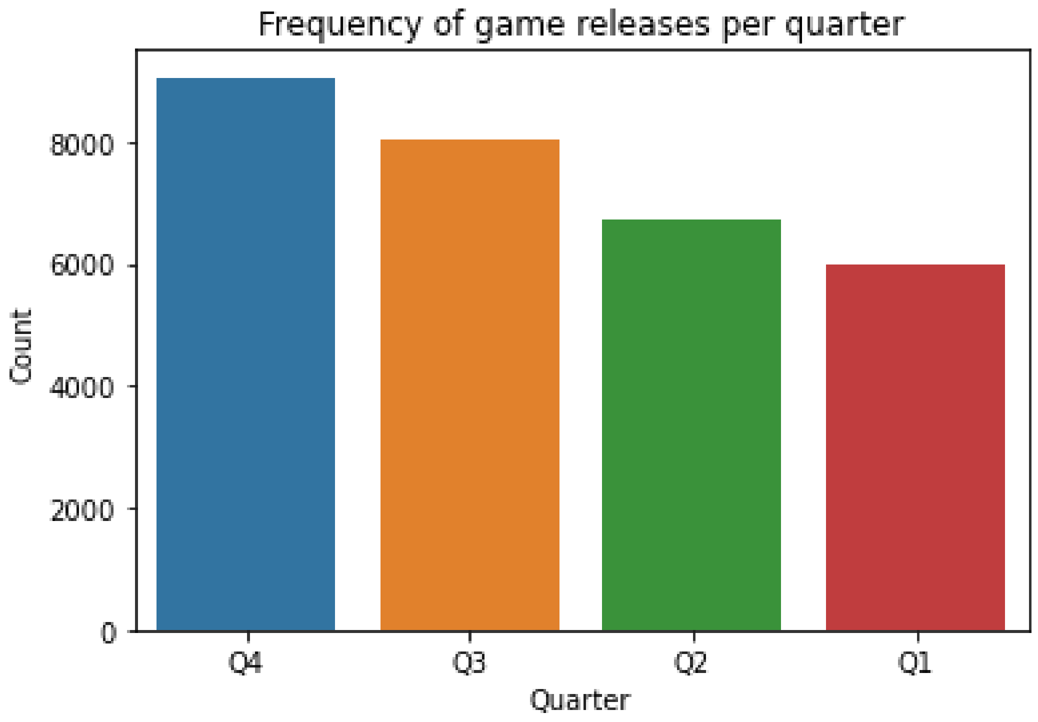
[ ]:
```
# Countplot of sale month

# define palette to highlight best months to buy house
custompalette = {release_month: "skyblue" if (release_month == 10 or␣
↪release_month == 11 or release_month == 12 ) else "lightgrey" \
                 for release_month in data['release_month'].unique()}

with sns.axes_style("whitegrid"):
    sns.countplot(x = data['release_month'], palette = custompalette, data =␣
↪data)
plt.title('Number of game releases per month')
plt.xlabel('Month')
plt.ylabel('Count')
plt.savefig('Images/month.pdf', bbox_inches = "tight")
```



We see that October, November and December have the highest number of game releases. Let's look at quarters now.

```
[ ]: # Define function to determine quarter
     def quarter(month):
         ''' Returns quarter in which month falls'''
         if 1 <= month <= 3:
             quarter = 'Q1'
         elif 4 <= month <= 6:
             quarter = 'Q2'
         elif 7 <= month <= 9:
             quarter = 'Q3'
         else:
             quarter = 'Q4'
         return quarter
```

```
[ ]: # Create quarter feature
     data['release_quarter'] = data['release_month'].apply(quarter)

     # Plot countplot using Seaborn
     sns.countplot(x = data['release_quarter'], data = data,
                   order = data['release_quarter'].value_counts().index)
     plt.title('Frequency of game releases per quarter')
     plt.xlabel('Quarter')
plt.ylabel('Count')
plt.show()
```



Recommendation:

Q4 and in particular the month of October sees the most new games released. We would recommend ensuring advertisement deals are priced at a premium during this period.

Finally, let's look at release date for the user-item data.

[ ]:
```
# Create copy to work with
releasedatedata = mergeddata.copy()

# Select entries where release date is not null
releasedatedata = releasedatedata[releasedatedata['release_date'].notnull()]

# Replace strings which are not of the format xxxx-xx-xx with None
releasedatedata['release_date'] = releasedatedata['release_date'].map(lambda x :
 ↪ x if x[-3] == '-'else None)

# Select entries where release date is not null
releasedatedata = releasedatedata[releasedatedata['release_date'].notnull()]

# Convert to DateTime
```

```
releasedatedata['release_date'] = pd.
 ↪to_datetime(releasedatedata['release_date'])

# Check
releasedatedata['release_date'].describe()
```

```
[ ]: count                    814101
     unique                     2598
     top       2012-08-21 00:00:00
     freq                       7086
     first     1983-06-19 00:00:00
     last      2018-12-01 00:00:00
     Name: release_date, dtype: object
```

Of course, we now have plenty of duplicate entries. However we note that the games span 1983 to 2018.

1.2.3 Game library size

```
[ ]: # View head
     numgames.head()
```

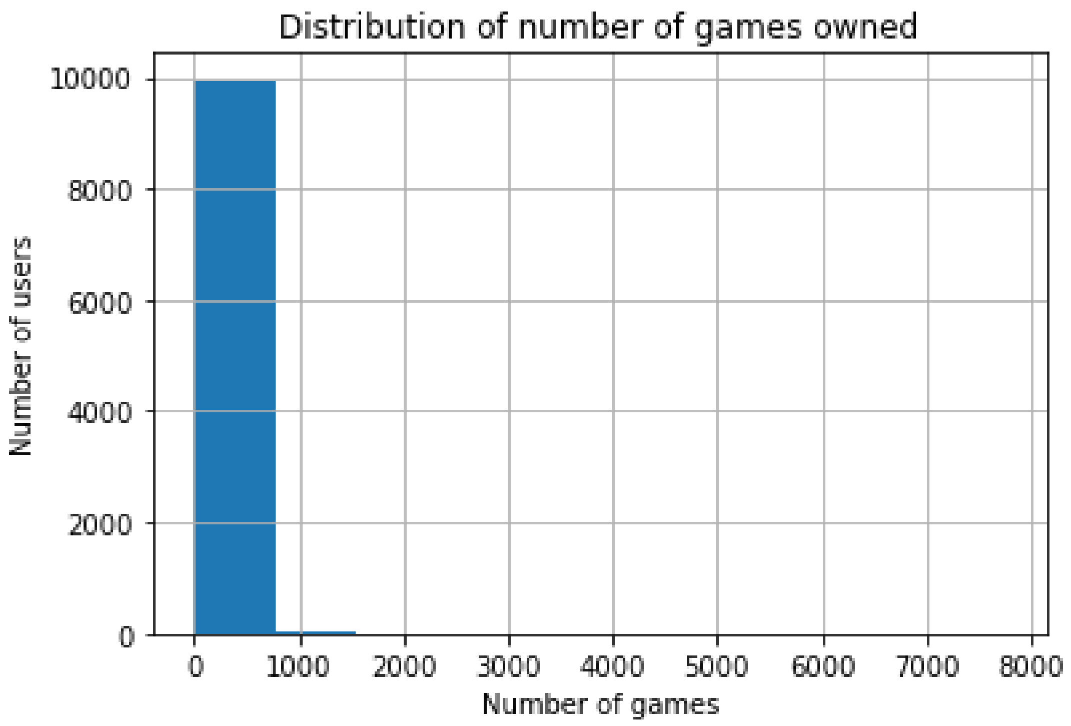
```
[ ]:              user_id  items_count
     0  76561197970982479          277
     1            js41637          888
     2          evcentric          137
     3         Riot-Punch          328
     4              doctr          541
```

```
[ ]: # Get summary statistics
     numgames['items_count'].describe()
```

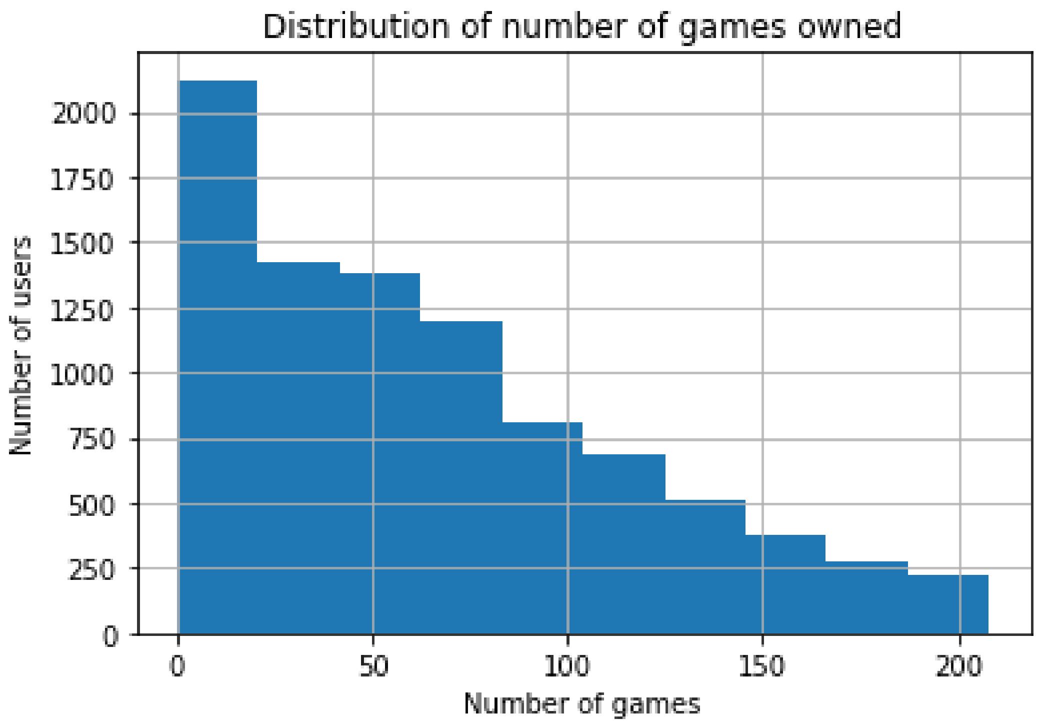
```
[ ]: count    10000.000000
     mean        99.498600
     std        194.502976
     min          0.000000
     25%         26.000000
     50%         64.000000
     75%        121.000000
     max       7762.000000
     Name: items_count, dtype: float64
```

We have data for 88310 unique steam users. We note that the minimum number of games owned is 0 whereas the maximum is 7762. The average number of games owned is 58.

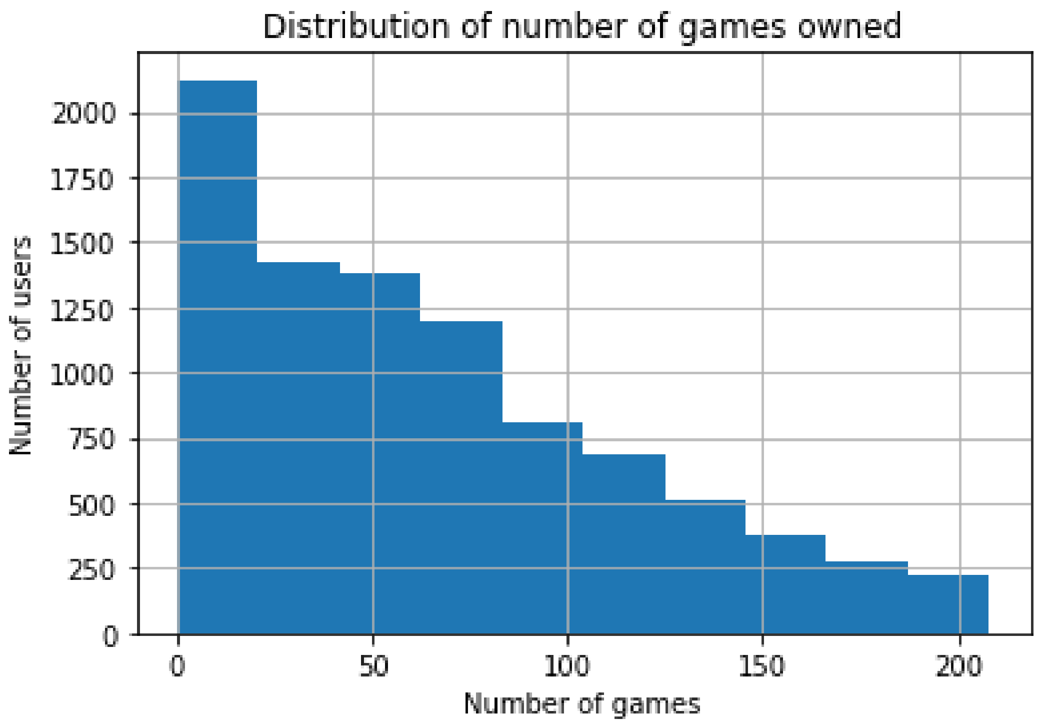
```
[ ]: # Plot distribution of `items_count`
     numgames['items_count'].hist()
     plt.title('Distribution of number of games owned')
plt.xlabel('Number of games')
plt.ylabel('Number of users')
plt.show()
```



[ ]:
```
# Plot distribution of items_count within 90% centile
numgames[numgames['items_count'] < numgames['items_count'].quantile(0.90)].
 ↪hist()
plt.title('Distribution of number of games owned')
plt.xlabel('Number of games')
plt.ylabel('Number of users')
plt.show()
```



```
[ ]: # Plot distribution of items_count within 90% centile
     numgames[numgames['items_count'] < numgames['items_count'].quantile(0.90)].
      ↪hist()
     plt.title('Distribution of number of games owned')
     plt.xlabel('Number of games')
     plt.ylabel('Number of users')
     plt.savefig('Images/numgames.pdf', bbox_inches = "tight")
     plt.show()
```



Recommendation:

Focus campaign on users who have below the average number of games of 58. These users are more likely to find games they do not own which appeal.

1.2.4 Game Price

```
[ ]: # Create a copy to work with
     gamesprice = gamesdata.copy()

     # Get statistics and type
     gamesprice['price'].describe()
```

```
[ ]: count      30758
     unique       162
     top         4.99
     freq        4278
     Name: price, dtype: object
```

We see that the values are of type `object`.

From viewing the head above, we noticed the presence of the string `Free To Play`. Let us replace that value with 0.

We will also iterate and replace all strings we find with 0.

[ ]:
```
gamesprice = gamesprice.replace(to_replace = 'Free To Play', value = 0)
gamesprice = gamesprice.replace(to_replace = 'Free to Play', value = 0)
gamesprice = gamesprice.replace(to_replace = 'Free', value = 0)
gamesprice = gamesprice.replace(to_replace = 'Free Demo', value = 0)
gamesprice = gamesprice.replace(to_replace = 'Play for Free!', value = 0)
gamesprice = gamesprice.replace(to_replace = 'Install Now', value = 0)
gamesprice = gamesprice.replace(to_replace = 'Play WARMACHINE: Tactics Demo',␣
 ↪value = 0)
gamesprice = gamesprice.replace(to_replace = 'Free Mod', value = 0)
gamesprice = gamesprice.replace(to_replace = 'Install Theme', value = 0)
gamesprice = gamesprice.replace(to_replace = 'Third-party', value = 0)
gamesprice = gamesprice.replace(to_replace = 'Play Now', value = 0)
gamesprice = gamesprice.replace(to_replace = 'Free HITMAN  Holiday Pack', value␣
 ↪= 0)
gamesprice = gamesprice.replace(to_replace = 'Play the Demo', value = 0)
gamesprice = gamesprice.replace(to_replace = 'Starting at $499.00', value = 0)
gamesprice = gamesprice.replace(to_replace = 'Starting at $449.00', value = 0)
gamesprice = gamesprice.replace(to_replace = 'Free to Try', value = 0)
gamesprice = gamesprice.replace(to_replace = 'Free Movie', value = 0)
gamesprice = gamesprice.replace(to_replace = 'Free to Use', value = 0)
```

[ ]:
```
gamesprice.price.unique()
```

[ ]:
```
array(['4.99', 0, '0.99', '2.99', '3.99', '9.99', '18.99', '29.99', nan,
       '10.99', '1.5899999999999999', '14.99', '1.99', '59.99', '8.99',
       '6.99', '7.99', '39.99', '19.99', '7.49', '12.99', '5.99', '2.49',
       '15.99', '1.25', '24.99', '17.99', '61.99', '3.49', '11.99',
       '13.99', '34.99', '74.76', '1.49', '32.99', '99.99', '14.95',
       '69.99', '16.99', '79.99', '49.99', '5.0', '44.99', '13.98',
       '29.96', '119.99', '109.99', '149.99', '771.71', '21.99', '89.99',
       '0.98', '139.92', '4.29', '64.99', '54.99', '74.99', '0.89', '0.5',
       '299.99', '1.29', '3.0', '15.0', '5.49', '23.99', '49.0', '20.99',
       '10.93', '1.3900000000000001', '36.99', '4.49', '2.0', '4.0',
       '9.0', '234.99', '1.9500000000000002', '1.5', '199.0', '189.0',
       '6.66', '27.99', '10.49', '129.99', '179.0', '26.99', '399.99',
       '31.99', '399.0', '20.0', '40.0', '3.33', '199.99', '22.99',
       '320.0', '38.85', '71.7', '59.95', '995.0', '27.49', '3.39', '6.0',
       '19.95', '499.99', '16.06', '4.68', '131.4', '44.98', '202.76',
       '1.0', '2.3', '0.9500000000000001', '172.24', '249.99',
       '2.9699999999999998', '10.96', '10.0', '30.0', '2.66', '6.48',
       '19.29', '11.15', '18.9', '2.89', '99.0', '87.94', '599.0', '8.98',
       '9.69', '0.49', '9.98', '9.95', '7.0', '12.89', '6.49', '1.87',
       '42.99', '41.99', '289.99', '23.96', '5.65', '12.0', '13.37',
       '189.96', '124.99', '19.98', '160.91'], dtype=object)
```

[ ]:
```
# Convert to float
gamesprice['price'] = gamesprice['price'].astype(float)
```

```
[ ]: # Get summary statistics
     gamesprice['price'].describe()
```

```
[ ]: count    30758.000000
     mean         8.866855
     std         15.903457
     min          0.000000
     25%          2.990000
     50%          4.990000
     75%          9.990000
     max        995.000000
     Name: price, dtype: float64
```

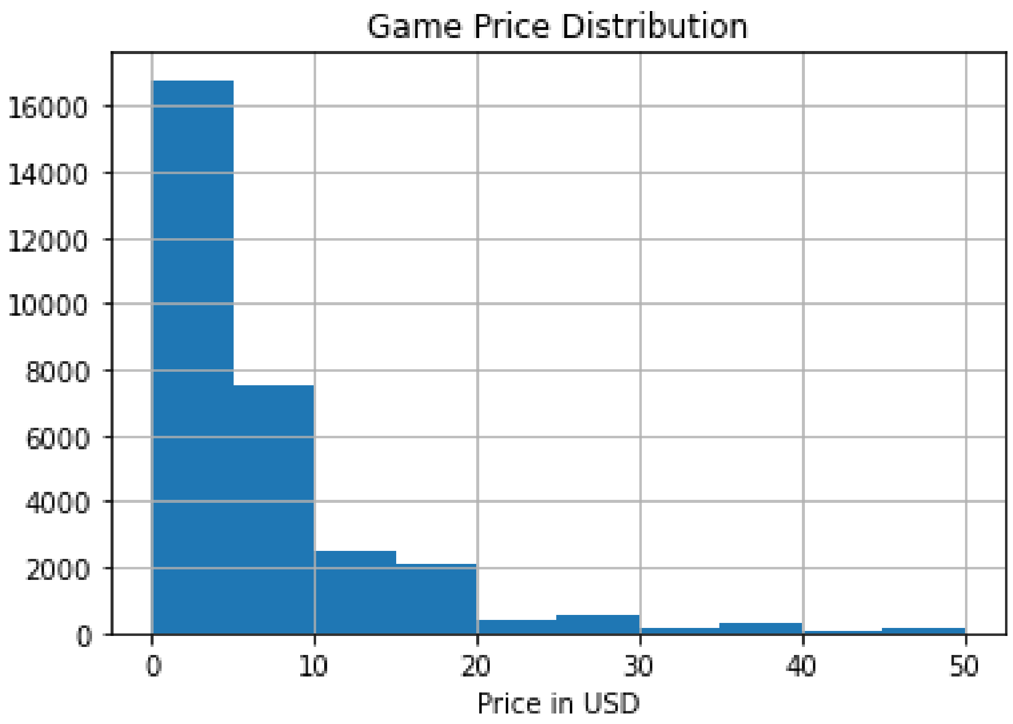
We see that 75% of games are under $10! Looks like the majority of games are cheap.

```
[ ]: belowcentile = gamesprice[gamesprice['price'] < gamesprice['price'].quantile(0.
      ↪99)]
```

```
[ ]: belowcentile['price'].describe()
```

```
[ ]: count    30440.000000
     mean         7.879879
     std          8.100161
     min          0.000000
     25%          2.990000
     50%          4.990000
     75%          9.990000
     max         49.990000
     Name: price, dtype: float64
```

```
[ ]: belowcentile['price'].hist()
     plt.xlabel('Price in USD')
     plt.title('Game Price Distribution')
     plt.savefig('Images/price.pdf', bbox_inches = "tight")
     plt.show()
```



Recommendation: Focus on volume of sales as the 75% of games are below $10. Highlights the importance of bundles for higher single transactions and where the user may not be interested in all games but still think it worthwhile.

1.2.5 Game genre

```
[ ]: gamesdata.head()
```

```
[ ]:          publisher                                                genres  \
     0        Kotoshiro  ['Action', 'Casual', 'Indie', 'Simulation', 'S…
     1  Making Fun, Inc.       ['Free to Play', 'Indie', 'RPG', 'Strategy']
     2      Poolians.com  ['Casual', 'Free to Play', 'Indie', 'Simulatio…
     3                              ['Action', 'Adventure', 'Casual']
     4              NaN                                               NaN

                    app_name                    title  \
     0      Lost Summoner Kitty      Lost Summoner Kitty
     1                Ironbound                Ironbound
     2  Real Pool 3D - Poolians  Real Pool 3D - Poolians
     3                    2222                    2222
     4            Log Challenge                      NaN

                                                      url release_date  \
     0  http://store.steampowered.com/app/761140/Lost_…   2018-01-04
```

```
1  http://store.steampowered.com/app/643980/Ironb…   2018-01-04
2  http://store.steampowered.com/app/670290/Real_…   2017-07-24
3     http://store.steampowered.com/app/767400/2222/   2017-12-07
4  http://store.steampowered.com/app/773570/Log_C…          NaN

                                                tags  discount_price  \
0  ['Strategy', 'Action', 'Indie', 'Casual', 'Sim…            4.49
1  ['Free to Play', 'Strategy', 'Indie', 'RPG', '…             NaN
2  ['Free to Play', 'Simulation', 'Sports', 'Casu…             NaN
3                 ['Action', 'Adventure', 'Casual']             0.83
4           ['Action', 'Indie', 'Casual', 'Sports']             1.79

                                        reviews_url  \
0  http://steamcommunity.com/app/761140/reviews/?…
1  http://steamcommunity.com/app/643980/reviews/?…
2  http://steamcommunity.com/app/670290/reviews/?…
3  http://steamcommunity.com/app/767400/reviews/?…
4  http://steamcommunity.com/app/773570/reviews/?…

                                              specs         price  \
0                                 ['Single-player']          4.99
1  ['Single-player', 'Multi-player', 'Online Mult…  Free To Play
2  ['Single-player', 'Multi-player', 'Online Mult…  Free to Play
3                                 ['Single-player']          0.99
4  ['Single-player', 'Full controller support', '…          2.99

   early_access        id         developer        sentiment  metascore
0         False  761140.0         Kotoshiro              NaN        NaN
1         False  643980.0  Secret Level SRL  Mostly Positive        NaN
2         False  670290.0      Poolians.com  Mostly Positive        NaN
3         False  767400.0                              NaN        NaN
4         False  773570.0               NaN              NaN        NaN
```

[ ]:
```
# Create copy
gamegenres = gamesdata.copy()

# Drop NaN
gamegenres = gamegenres[gamegenres['genres'].notnull()]

# Get unique lists
genres = list(gamegenres['genres'].unique())

# View first 5
genres[:5]
```

[ ]:
```
["['Action', 'Casual', 'Indie', 'Simulation', 'Strategy']",
 "['Free to Play', 'Indie', 'RPG', 'Strategy']",
 "['Casual', 'Free to Play', 'Indie', 'Simulation', 'Sports']",
 "['Action', 'Adventure', 'Casual']",
 "['Action', 'Adventure', 'Simulation']"]
```

```
[ ]: # Combine all strings
     allgenres = ','.join(genres)

     # Preview first 100 characters
     allgenres[:100]
```

```
[ ]: "['Action', 'Casual', 'Indie', 'Simulation', 'Strategy'],['Free to Play',
     'Indie', 'RPG', 'Strategy']"
```

```
[ ]: # Replace chars
     allgenres = allgenres.replace("[","").replace("]", "").replace("'", "").
      ↪replace(" ","")

     # Check
     allgenres[:100]
```

```
[ ]: 'Action,Casual,Indie,Simulation,Strategy,FreetoPlay,Indie,RPG,Strategy,Casual,Fr
     eetoPlay,Indie,Simula'
```

```
[ ]: # Split
     splitgenres = allgenres.split(',')
     splitgenres[:5]
```

```
[ ]: ['Action', 'Casual', 'Indie', 'Simulation', 'Strategy']
```

```
[ ]: # Use set to obtain unique values
     uniquegenres = set(splitgenres)
     uniquegenres
```

```
[ ]: {'Accounting',
      'Action',
      'Adventure',
      'Animation&amp;Modeling',
      'AudioProduction',
      'Casual',
      'Design&amp;Illustration',
      'EarlyAccess',
      'Education',
      'FreetoPlay',
      'Indie',
      'MassivelyMultiplayer',
      'PhotoEditing',
      'RPG',
      'Racing',
 'Simulation',
 'SoftwareTraining',
 'Sports',
 'Strategy',
 'Utilities',
 'VideoProduction',
 'WebPublishing'}
```

[ ]:
```
# Create columns with genres
for genre in uniquegenres:
    gamegenres[genre] = 0

# Split genres in genres column
gamegenres['genres'] = gamegenres['genres'].map(lambda x : x.replace("["," ").
 ↪replace("]", "").replace("'", "").replace(" ","").split(','))

# Map to columns - set to 1 if genre applies
for index, genres in enumerate(gamegenres['genres']):
    for genre in genres:
        gamegenres.loc[index,genre] = 1

# Visuale the new columns
gamegenres.head(2)
```

[ ]:
```
          publisher                                          genres  \
0         Kotoshiro  [Action, Casual, Indie, Simulation, Strategy]
1  Making Fun, Inc.              [FreetoPlay, Indie, RPG, Strategy]

              app_name                title  \
0  Lost Summoner Kitty  Lost Summoner Kitty
1            Ironbound            Ironbound

                                                 url release_date  \
0  http://store.steampowered.com/app/761140/Lost_…   2018-01-04
1  http://store.steampowered.com/app/643980/Ironb…   2018-01-04

                                                tags  discount_price  \
0  ['Strategy', 'Action', 'Indie', 'Casual', 'Sim…            4.49
1  ['Free to Play', 'Strategy', 'Indie', 'RPG', '…             NaN

                                         reviews_url  \
0  http://steamcommunity.com/app/761140/reviews/?…
1  http://steamcommunity.com/app/643980/reviews/?…

                                               specs  …  \
0                                  ['Single-player']  …
1  ['Single-player', 'Multi-player', 'Online Mult…  …
```

```
  Animation&amp;Modeling Racing  Casual  Education EarlyAccess  Utilities  \
0                     0.0     0.0     1.0       0.0         0.0        0.0
1                     0.0     0.0     0.0       0.0         0.0        0.0

   WebPublishing  PhotoEditing  Design&amp;Illustration  Indie
0            0.0           0.0                     0.0    1.0
1            0.0           0.0                     0.0    1.0

[2 rows x 38 columns]
```

[ ]:
```
gamegenres.columns
```

[ ]:
```
Index(['publisher', 'genres', 'app_name', 'title', 'url', 'release_date',
       'tags', 'discount_price', 'reviews_url', 'specs', 'price',
       'early_access', 'id', 'developer', 'sentiment', 'metascore',
       'Adventure', 'Sports', 'RPG', 'FreetoPlay', 'SoftwareTraining',
       'VideoProduction', 'AudioProduction', 'Simulation', 'Action',
       'Strategy', 'MassivelyMultiplayer', 'Accounting',
       'Animation&amp;Modeling', 'Racing', 'Casual', 'Education',
       'EarlyAccess', 'Utilities', 'WebPublishing', 'PhotoEditing',
       'Design&amp;Illustration', 'Indie'],
      dtype='object')
```

[ ]:
```
# Start with empty dictionary
genredict = {}

# Get genre columns
genrecols = gamegenres.loc[:, 'Adventure':'Indie'].columns

# Go through each column and sum it
for col in genrecols:
    genredict[col] = gamegenres[col].sum()

# sort dictionary based on counts, ascending order so reverse = True
sortedgenresdict = {keys: values for keys, values in \
                        sorted(genredict.items(), key = lambda item: item[1],␣
 ↪reverse = True)}
```

[ ]:
```
# View dictionary
sortedgenresdict
```

[ ]:
```
{'Indie': 15858.0,
 'Action': 11321.0,
 'Casual': 8282.0,
 'Adventure': 8243.0,
 'Strategy': 6957.0,
 'Simulation': 6699.0,
 'RPG': 5479.0,
 'FreetoPlay': 2031.0,
 'EarlyAccess': 1462.0,
 'Sports': 1257.0,
 'MassivelyMultiplayer': 1108.0,
 'Racing': 1083.0,
 'Design&amp;Illustration': 460.0,
 'Utilities': 340.0,
 'WebPublishing': 268.0,
 'Animation&amp;Modeling': 183.0,
 'Education': 125.0,
 'VideoProduction': 116.0,
 'SoftwareTraining': 105.0,
 'AudioProduction': 93.0,
 'PhotoEditing': 77.0,
 'Accounting': 7.0}
```

We see that Indie is the most popular genre, followed by Action. On the other end of the spectrum, there are few entries relating to Photo Editing and only 7 for Accounting. This makes sense as Steam is a gaming platform, and so photo editing or accounting software doesn't really belong.

1.2.6 Game tags

[ ]:
```python
# Create copy
gametags = gamesdata.copy()

# Drop NaN
gametags = gamegenres[gamegenres['tags'].notnull()]

# Get unique lists
tags = list(gametags['tags'].unique())

# View first 5
tags[:5]
```

[ ]:
```
["['Strategy', 'Action', 'Indie', 'Casual', 'Simulation']",
 "['Free to Play', 'Strategy', 'Indie', 'RPG', 'Card Game', 'Trading Card Game',
'Turn-Based', 'Fantasy', 'Tactical', 'Dark Fantasy', 'Board Game', 'PvP', '2D',
'Competitive', 'Replay Value', 'Character Customization', 'Female Protagonist',
'Difficult', 'Design & Illustration']",
 "['Free to Play', 'Simulation', 'Sports', 'Casual', 'Indie', 'Multiplayer']",
 "['Action', 'Adventure', 'Casual']",
 "['Action', 'Adventure', 'Simulation', 'FPS', 'Shooter', 'Third-Person
Shooter', 'Sniper', 'Third Person']"]
```

```
[ ]: # Combine all strings
     alltags = ','.join(tags)

     # Preview first 100 characters
     alltags[:100]
```

```
[ ]: "['Strategy', 'Action', 'Indie', 'Casual', 'Simulation'],['Free to Play',
     'Strategy', 'Indie', 'RPG',"
```

```
[ ]: # Replace chars
     alltags = alltags.replace("["," ").replace("]", "").replace("'", "")

     # Check
     alltags[:100]
```

```
[ ]: ' Strategy, Action, Indie, Casual, Simulation, Free to Play, Strategy, Indie,
     RPG, Card Game, Trading'
```

```
[ ]: # Split
     splittags = alltags[1:].split(',')
     splittags[:5]
```

```
[ ]: ['Strategy', ' Action', ' Indie', ' Casual', ' Simulation']
```

```
[ ]: # Use set to obtain unique values
     uniquetags = set(splittags)
     len(uniquetags)
```

```
[ ]: 337
```

1.2.7 Top publishers

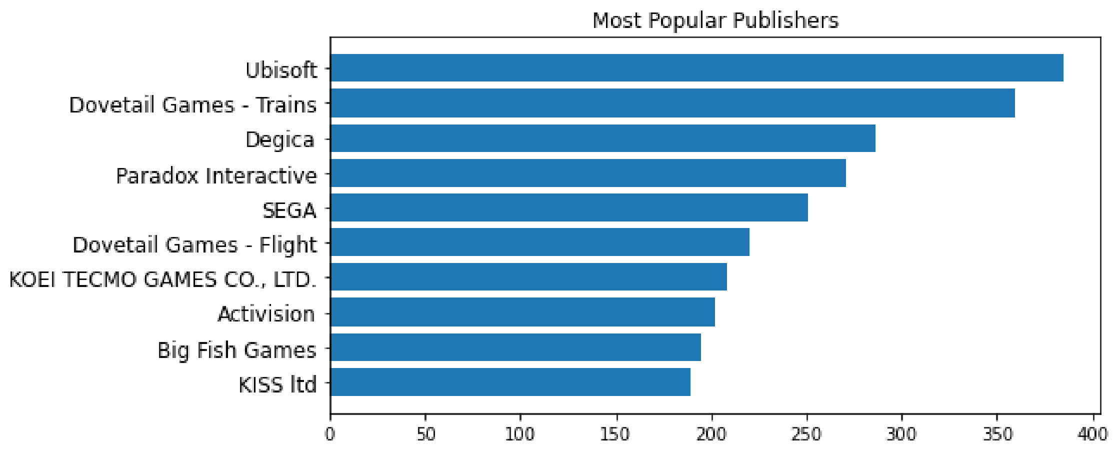
```
[ ]: # Select entries where publisher is non-null
     data = gamesdata[gamesdata['publisher'].notnull()]
```

```
[ ]: # Create dictionary
     game_publishers = {}
     for publisher in list(data['publisher']):
         if not publisher in game_publishers:
             game_publishers[publisher] = 1
         else:
             game_publishers[publisher] += 1
```

```
[ ]: # Get top 10 publishers
     top10_publishers = dict(Counter(game_publishers).most_common(10))
     top10_publishers
```

[ ]:
```
{'Ubisoft': 385,
 'Dovetail Games - Trains': 360,
 'Degica': 286,
 'Paradox Interactive': 271,
 'SEGA': 251,
 'Dovetail Games - Flight': 220,
 'KOEI TECMO GAMES CO., LTD.': 208,
 'Activision': 202,
 'Big Fish Games': 195,
 'KISS ltd': 189}
```

[ ]:
```
# Prepare for bar chart plot
top10_publishers = dict(sorted(Counter(game_publishers).most_common(10),␣
 ↪key=lambda x:x[1]))

# Plots most popular publishers
fig = plt.figure(figsize = (8,4))
plt.barh(range(len(top10_publishers)), list(top10_publishers.values()),␣
 ↪align='center')
plt.yticks(range(len(top10_publishers)), list(top10_publishers.keys()),␣
 ↪fontsize=12)
plt.title("Most Popular Publishers", fontsize=12, fontweight= 22)
plt.show()
```



[ ]: